Dicionários e conjuntos

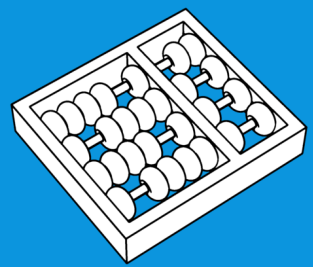
MC102 - Algoritmos e Programação de Computadores

04/25

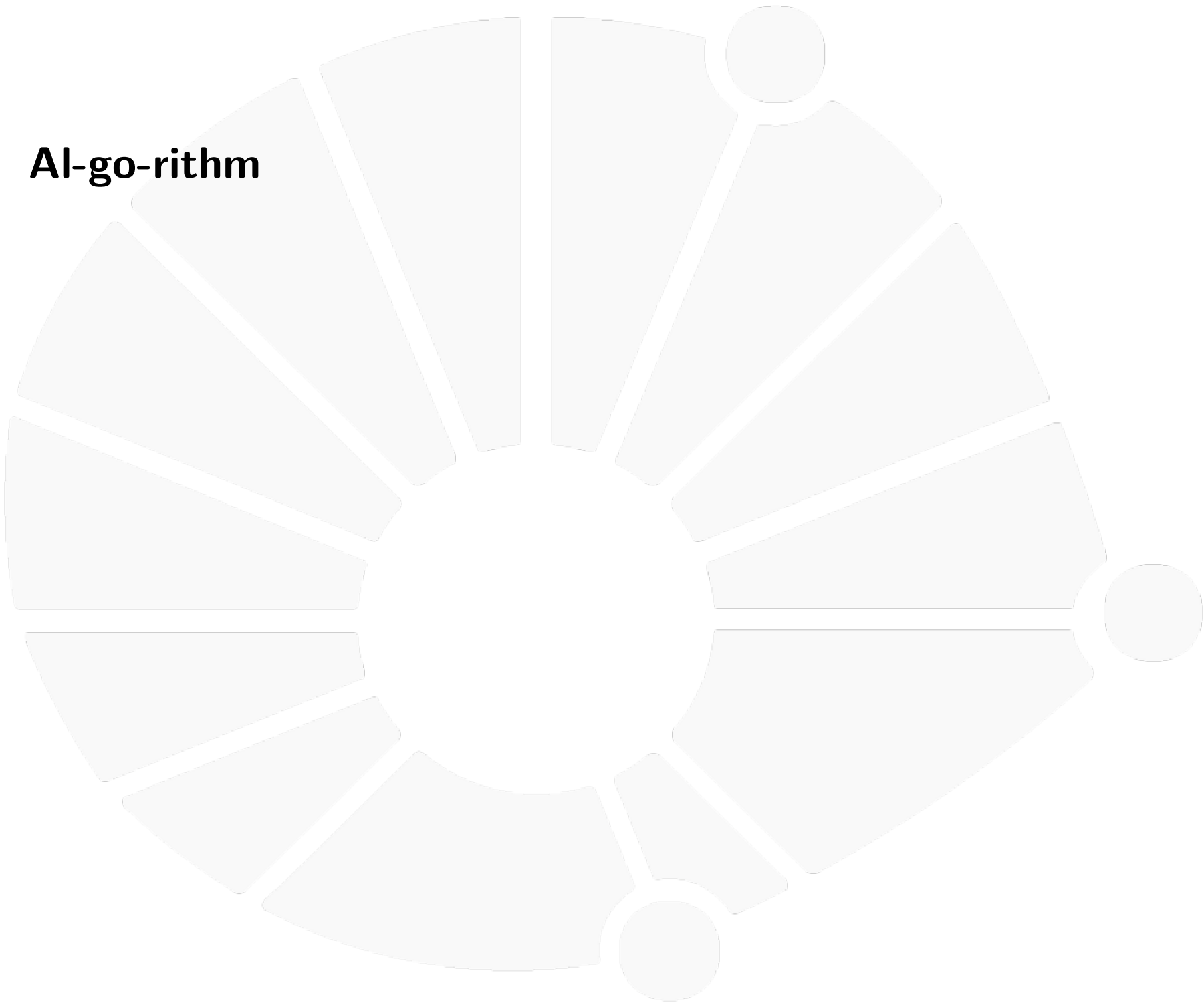
14

Santiago Valdés Ravelo
https://ic.unicamp.br/~santiago/
ravelo@unicamp.br

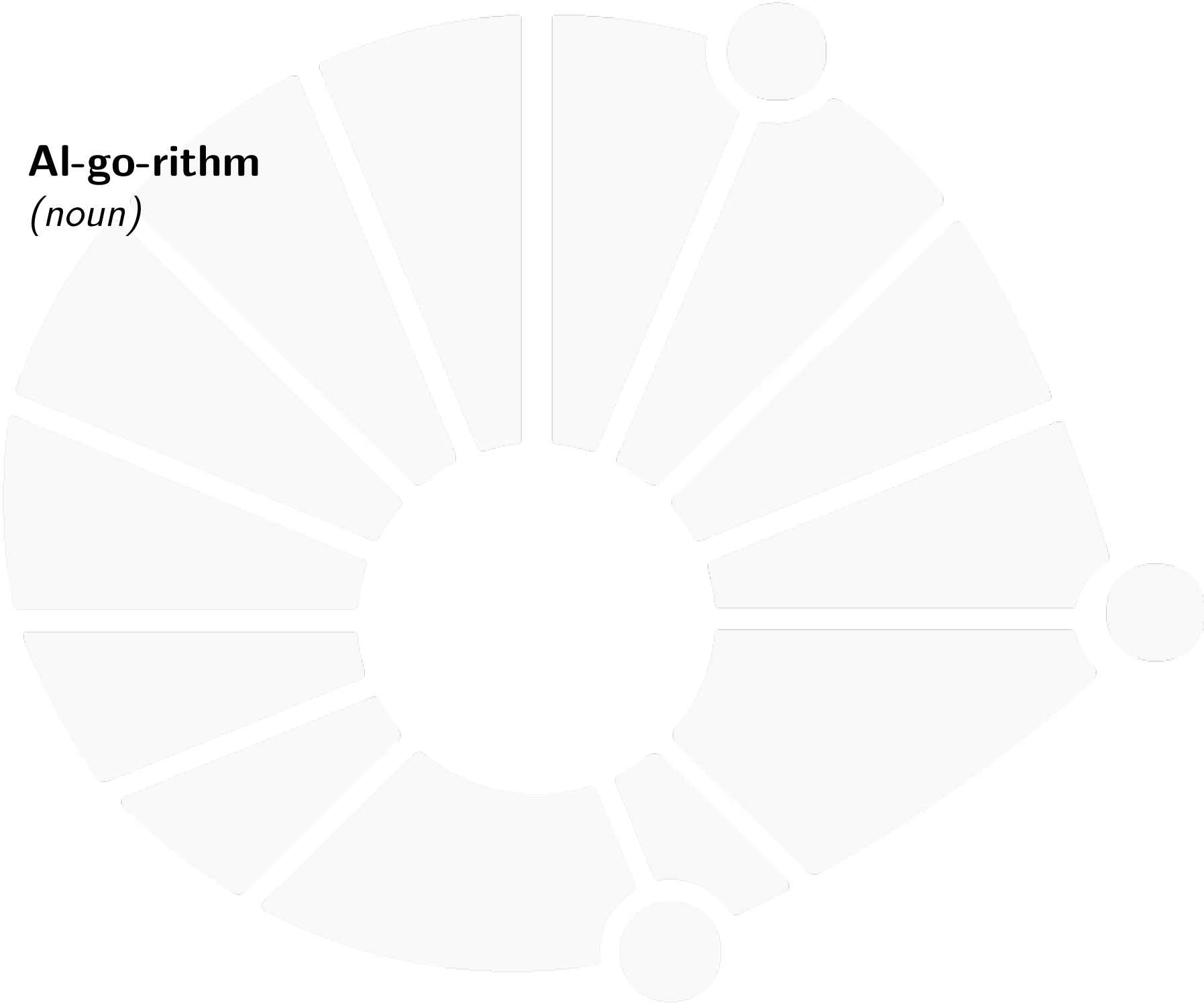
UNICAMP



Al-go-rithm

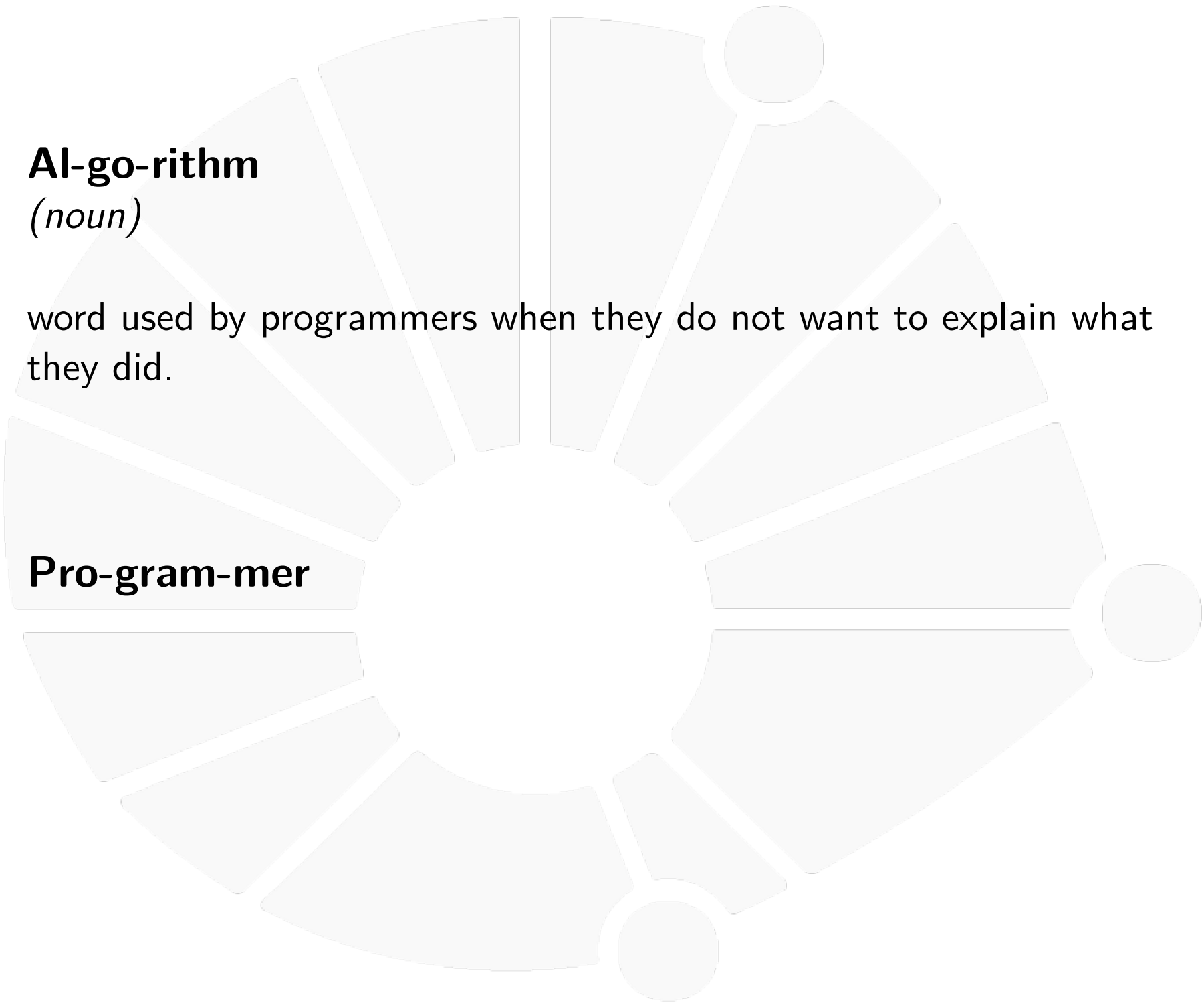


Al-go-rithm
(noun)



Al-go-rithm
(noun)

word used by programmers when they do not want to explain what they did.



Al-go-rithm
(noun)

word used by programmers when they do not want to explain what they did.

Pro-gram-mer

Al-go-rithm
(noun)

word used by programmers when they do not want to explain what they did.

Pro-gram-mer
(noun)

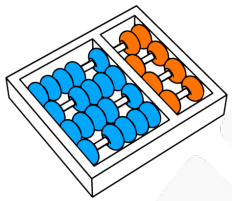
Al-go-rithm
(noun)

word used by programmers when they do not want to explain what they did.

Pro-gram-mer
(noun)

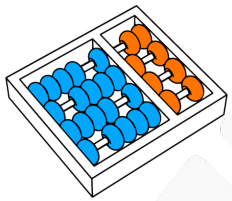
someone who solves a problem you didn't know you had in a way you don't understand.

Dúvidas da aula anterior



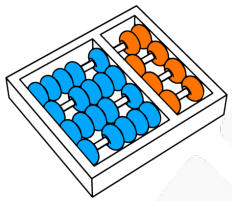
Dúvidas selecionadas

- Uma string é a mesma coisa que uma lista de caracteres ou são coisas diferentes?



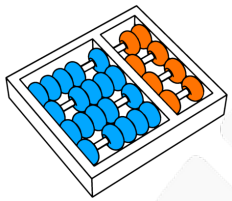
Dúvidas selecionadas

- Uma string é a mesma coisa que uma lista de caracteres ou são coisas diferentes?
- Dada uma lista l de strings, l[0][0] acessa o primeiro caractere da primeira string?



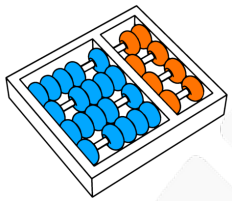
Dúvidas selecionadas

- Uma string é a mesma coisa que uma lista de caracteres ou são coisas diferentes?
- Dada uma lista l de strings, l[0][0] acessa o primeiro caractere da primeira string?
- str[x:y:z] seria igual ao range(x,y,z) em relação ao início, fim e passo?



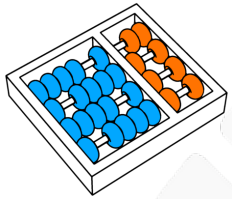
Dúvidas selecionadas

- Uma string é a mesma coisa que uma lista de caracteres ou são coisas diferentes?
- Dada uma lista l de strings, l[0][0] acessa o primeiro caractere da primeira string?
- str[x:y:z] seria igual ao range(x,y,z) em relação ao início, fim e passo?
- Por que o padrão python não quis criar um tipo caracter? Não simplificaria operações como string[i] = char? Seria só construir um tipo string baseado em uma lista de char.



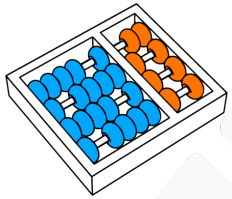
Dúvidas selecionadas

- Uma string é a mesma coisa que uma lista de caracteres ou são coisas diferentes?
- Dada uma lista l de strings, l[0][0] acessa o primeiro caractere da primeira string?
- str[x:y:z] seria igual ao range(x,y,z) em relação ao início, fim e passo?
- Por que o padrão python não quis criar um tipo caracter? Não simplificaria operações como string[i] = char? Seria só construir um tipo string baseado em uma lista de char.
- Tendo em vista que o python não reconhece a barra invertida \ sozinha como um caracter (ela precisa associar-se a outro caracter para formar um caracter especial), o que acontece se eu armazenar numa variável s, via input, a string "\n", por exemplo? E ainda, se eu tentar imprimir essa variável s, ele vai imprimir "\n" ou vai imprimir uma quebra de linha?



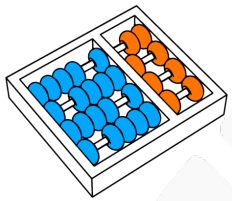
Dúvidas selecionadas

- Uma string é a mesma coisa que uma lista de caracteres ou são coisas diferentes?
- Dada uma lista l de strings, l[0][0] acessa o primeiro caractere da primeira string?
- str[x:y:z] seria igual ao range(x,y,z) em relação ao início, fim e passo?
- Por que o padrão python não quis criar um tipo caracter? Não simplificaria operações como string[i] = char? Seria só construir um tipo string baseado em uma lista de char.
- Tendo em vista que o python não reconhece a barra invertida \ sozinha como um caracter (ela precisa associar-se a outro caracter para formar um caracter especial), o que acontece se eu armazenar numa variável s, via input, a string "\n", por exemplo? E ainda, se eu tentar imprimir essa variável s, ele vai imprimir "\n" ou vai imprimir uma quebra de linha?
- Não entendi muito bem como usa o .join()



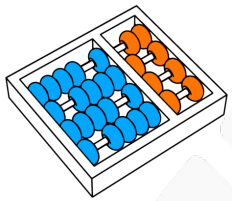
Dúvidas selecionadas

- Uma string é a mesma coisa que uma lista de caracteres ou são coisas diferentes?
- Dada uma lista l de strings, l[0][0] acessa o primeiro caractere da primeira string?
- str[x:y:z] seria igual ao range(x,y,z) em relação ao início, fim e passo?
- Por que o padrão python não quis criar um tipo caracter? Não simplificaria operações como string[i] = char? Seria só construir um tipo string baseado em uma lista de char.
- Tendo em vista que o python não reconhece a barra invertida \ sozinha como um caracter (ela precisa associar-se a outro caracter para formar um caracter especial), o que acontece se eu armazenar numa variável s, via input, a string "\n", por exemplo? E ainda, se eu tentar imprimir essa variável s, ele vai imprimir "\n" ou vai imprimir uma quebra de linha?
- Não entendi muito bem como usa o .join()
- Não entendi por que eu usaria o format da forma dos exemplos de "format mais legível".



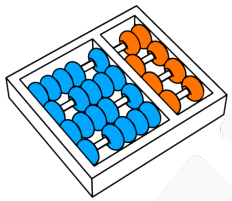
Dúvidas selecionadas

- Uma string é a mesma coisa que uma lista de caracteres ou são coisas diferentes?
- Dada uma lista l de strings, l[0][0] acessa o primeiro caractere da primeira string?
- str[x:y:z] seria igual ao range(x,y,z) em relação ao início, fim e passo?
- Por que o padrão python não quis criar um tipo caracter? Não simplificaria operações como string[i] = char? Seria só construir um tipo string baseado em uma lista de char.
- Tendo em vista que o python não reconhece a barra invertida \ sozinha como um caracter (ela precisa associar-se a outro caracter para formar um caracter especial), o que acontece se eu armazenar numa variável s, via input, a string "\n", por exemplo? E ainda, se eu tentar imprimir essa variável s, ele vai imprimir "\n" ou vai imprimir uma quebra de linha?
- Não entendi muito bem como usa o .join()
- Não entendi por que eu usaria o format da forma dos exemplos de "format mais legível".
- Não entendi muito bem a parte de usar \ no código.



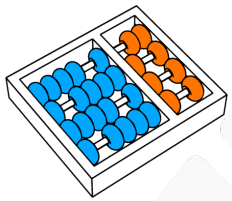
Dúvidas selecionadas

- Uma string é a mesma coisa que uma lista de caracteres ou são coisas diferentes?
- Dada uma lista l de strings, l[0][0] acessa o primeiro caractere da primeira string?
- str[x:y:z] seria igual ao range(x,y,z) em relação ao início, fim e passo?
- Por que o padrão python não quis criar um tipo caracter? Não simplificaria operações como string[i] = char? Seria só construir um tipo string baseado em uma lista de char.
- Tendo em vista que o python não reconhece a barra invertida \ sozinha como um caracter (ela precisa associar-se a outro caracter para formar um caracter especial), o que acontece se eu armazenar numa variável s, via input, a string "\n", por exemplo? E ainda, se eu tentar imprimir essa variável s, ele vai imprimir "\n" ou vai imprimir uma quebra de linha?
- Não entendi muito bem como usa o .join()
- Não entendi por que eu usaria o format da forma dos exemplos de "format mais legível".
- Não entendi muito bem a parte de usar \ no código.
- Como lidar/tratar strings com espaços vazios ou com um espaço vazio ao final? Por excemplo: Se o usuário digita um espaço a mais no final e eu preciso validar o resultado.



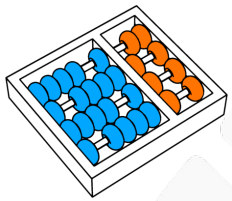
Dúvidas selecionadas

- Uma string é a mesma coisa que uma lista de caracteres ou são coisas diferentes?
- Dada uma lista l de strings, l[0][0] acessa o primeiro caractere da primeira string?
- str[x:y:z] seria igual ao range(x,y,z) em relação ao início, fim e passo?
- Por que o padrão python não quis criar um tipo caracter? Não simplificaria operações como string[i] = char? Seria só construir um tipo string baseado em uma lista de char.
- Tendo em vista que o python não reconhece a barra invertida \ sozinha como um caracter (ela precisa associar-se a outro caracter para formar um caracter especial), o que acontece se eu armazenar numa variável s, via input, a string "\n", por exemplo? E ainda, se eu tentar imprimir essa variável s, ele vai imprimir "\n" ou vai imprimir uma quebra de linha?
- Não entendi muito bem como usa o .join()
- Não entendi por que eu usaria o format da forma dos exemplos de "format mais legível".
- Não entendi muito bem a parte de usar \ no código.
- Como lidar/tratar strings com espaços vazios ou com um espaço vazio ao final? Por excemplo: Se o usuário digita um espaço a mais no final e eu preciso validar o resultado.
- Porque eu não posso modificar uma string inplace (ex: texto[0] = 'o')?



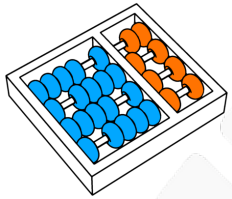
Dúvidas selecionadas

- Uma string é a mesma coisa que uma lista de caracteres ou são coisas diferentes?
- Dada uma lista l de strings, l[0][0] acessa o primeiro caractere da primeira string?
- str[x:y:z] seria igual ao range(x,y,z) em relação ao início, fim e passo?
- Por que o padrão python não quis criar um tipo caracter? Não simplificaria operações como string[i] = char? Seria só construir um tipo string baseado em uma lista de char.
- Tendo em vista que o python não reconhece a barra invertida \ sozinha como um caracter (ela precisa associar-se a outro caracter para formar um caracter especial), o que acontece se eu armazenar numa variável s, via input, a string "\n", por exemplo? E ainda, se eu tentar imprimir essa variável s, ele vai imprimir "\n" ou vai imprimir uma quebra de linha?
- Não entendi muito bem como usa o .join()
- Não entendi por que eu usaria o format da forma dos exemplos de "format mais legível".
- Não entendi muito bem a parte de usar \ no código.
- Como lidar/tratar strings com espaços vazios ou com um espaço vazio ao final? Por excemplo: Se o usuário digita um espaço a mais no final e eu preciso validar o resultado.
- Porque eu não posso modificar uma string inplace (ex: texto[0] = 'o')?
- Formatando strings, dá pra usar múltiplos formatadores como .2f e % no mesmo valor? Algo como f"{valor:%.2f}" ou coisa parecida, que faria o número 0.300000009 ser mostrado como "30.00%"



Dúvidas selecionadas

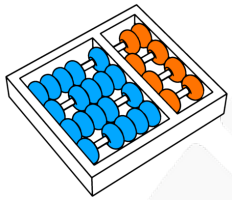
- Uma string é a mesma coisa que uma lista de caracteres ou são coisas diferentes?
- Dada uma lista l de strings, l[0][0] acessa o primeiro caractere da primeira string?
- str[x:y:z] seria igual ao range(x,y,z) em relação ao início, fim e passo?
- Por que o padrão python não quis criar um tipo caracter? Não simplificaria operações como string[i] = char? Seria só construir um tipo string baseado em uma lista de char.
- Tendo em vista que o python não reconhece a barra invertida \ sozinha como um caracter (ela precisa associar-se a outro caracter para formar um caracter especial), o que acontece se eu armazenar numa variável s, via input, a string "\n", por exemplo? E ainda, se eu tentar imprimir essa variável s, ele vai imprimir "\n" ou vai imprimir uma quebra de linha?
- Não entendi muito bem como usa o .join()
- Não entendi por que eu usaria o format da forma dos exemplos de "format mais legível".
- Não entendi muito bem a parte de usar \ no código.
- Como lidar/tratar strings com espaços vazios ou com um espaço vazio ao final? Por excemplo: Se o usuário digita um espaço a mais no final e eu preciso validar o resultado.
- Porque eu não posso modificar uma string inplace (ex: texto[0] = 'o')?
- Formatando strings, dá pra usar múltiplos formatadores como .2f e % no mesmo valor? Algo como f"{valor:%.2f}" ou coisa parecida, que faria o número 0.300000009 ser mostrado como "30.00%"
- Com o método .find('a') eu recebo apenas o índice da primeira aparição de a? Se sim teria uma forma de obter as demais?



Dúvidas selecionadas

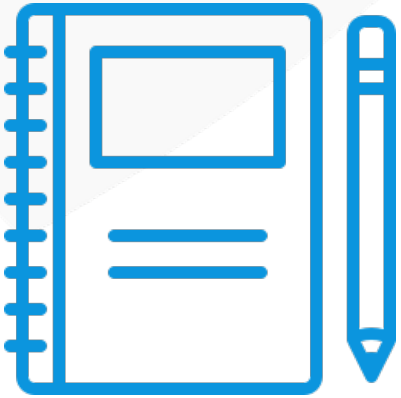
- Uma string é a mesma coisa que uma lista de caracteres ou são coisas diferentes?
- Dada uma lista l de strings, l[0][0] acessa o primeiro caractere da primeira string?
- str[x:y:z] seria igual ao range(x,y,z) em relação ao início, fim e passo?
- Por que o padrão python não quis criar um tipo caracter? Não simplificaria operações como string[i] = char? Seria só construir um tipo string baseado em uma lista de char.
- Tendo em vista que o python não reconhece a barra invertida \ sozinha como um caracter (ela precisa associar-se a outro caracter para formar um caracter especial), o que acontece se eu armazenar numa variável s, via input, a string "\n", por exemplo? E ainda, se eu tentar imprimir essa variável s, ele vai imprimir "\n" ou vai imprimir uma quebra de linha?
- Não entendi muito bem como usa o .join()
- Não entendi por que eu usaria o format da forma dos exemplos de "format mais legível".
- Não entendi muito bem a parte de usar \ no código.
- Como lidar/tratar strings com espaços vazios ou com um espaço vazio ao final? Por excemplo: Se o usuário digita um espaço a mais no final e eu preciso validar o resultado.
- Porque eu não posso modificar uma string inplace (ex: texto[0] = 'o')?
- Formatando strings, dá pra usar múltiplos formatadores como .2f e % no mesmo valor? Algo como f"{valor:%.2f}" ou coisa parecida, que faria o número 0.300000009 ser mostrado como "30.00%"
- Com o método .find('a') eu recebo apenas o índice da primeira aparição de a? Se sim teria uma forma de obter as demais?
- O que acontece se eu por '\\\' na string?

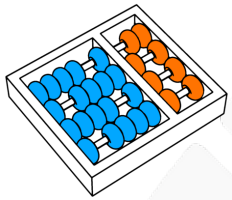
Motivação



Exercícios

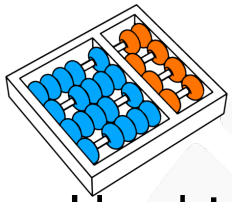
Existe motivação melhor que alguns exercícios?





Exercício 1

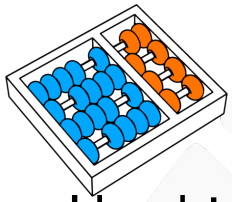
1. Leia uma quantidade **n** de pares de nome/RA de alunos. Faça um programa que permita buscar o RA de um aluno a partir do seu nome.



Exercícios

Um histograma é uma representação de dados utilizado na estatística. Simplificando, a ideia é exibir a quantidade de vezes que um certo dado aparece. Exemplo:

```
0:  ***
1:  *****
2:  *
3:
4:  ********
5:  **
6:  ****
7:  *****
8:
9:  **
```

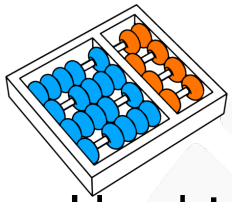


Exercícios

Um histograma é uma representação de dados utilizado na estatística. Simplificando, a ideia é exibir a quantidade de vezes que um certo dado aparece. Exemplo:

```
0:  ***
1:  *****
2:  *
3:
4:  ********
5:  **
6:  ****
7:  *****
8:
9:  **
```

2 Faça um programa, que dado uma lista de número inteiros em $[0, 10)$, imprime um histograma para tais números.



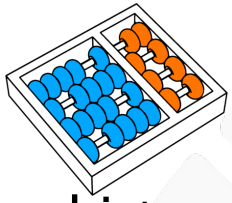
Exercícios

Um histograma é uma representação de dados utilizado na estatística. Simplificando, a ideia é exibir a quantidade de vezes que um certo dado aparece. Exemplo:

```
0:  ***
1:  *****
2:  *
3:
4:  ********
5:  **
6:  ****
7:  *****
8:
9:  **
```

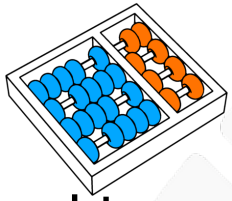
2. Faça um programa, que dado uma lista de número inteiros em $[0, 10)$, imprime um histograma para tais números.
3. Como modificar o programa anterior para fazer um histograma de nomes de alunos?

Dicionários



Dicionários

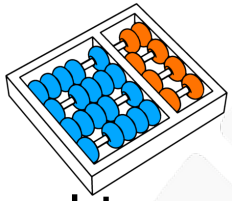
Listas armazenam os dados de acordo com os índices:



Dicionários

Listas armazenam os dados de acordo com os índices:

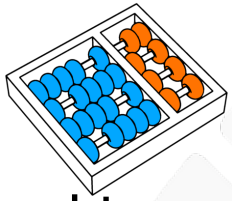
- Os dados são acessados pelo índice:



Dicionários

Listas armazenam os dados de acordo com os índices:

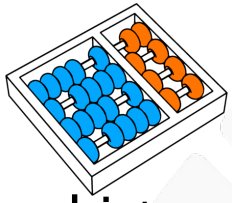
- Os dados são acessados pelo índice:
  - Um **int**.



Dicionários

Listas armazenam os dados de acordo com os índices:

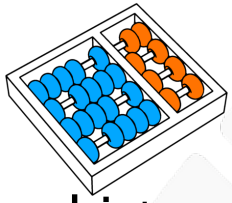
- Os dados são acessados pelo índice:
  - Um **int**.
- Para cada índice, temos um valor:



Dicionários

Listas armazenam os dados de acordo com os índices:

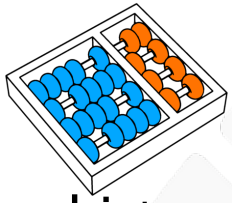
- Os dados são acessados pelo índice:
  - Um **int**.
- Para cada índice, temos um valor:
  - É uma função no sentido matemático.



Dicionários

Listas armazenam os dados de acordo com os índices:

- Os dados são acessados pelo índice:
  - Um **int**.
- Para cada índice, temos um valor:
  - É uma função no sentido matemático.
- Podemos pensar que os nossos dados são pares índice-valor.

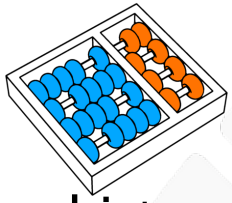


Dicionários

Listas armazenam os dados de acordo com os índices:

- Os dados são acessados pelo índice:
  - Um **int**.
- Para cada índice, temos um valor:
  - É uma função no sentido matemático.
- Podemos pensar que os nossos dados são pares índice-valor.

Dicionários armazenam os dados em pares chave-valor:



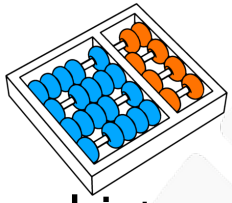
Dicionários

Listas armazenam os dados de acordo com os índices:

- Os dados são acessados pelo índice:
  - Um **int**.
- Para cada índice, temos um valor:
  - É uma função no sentido matemático.
- Podemos pensar que os nossos dados são pares índice-valor.

Dicionários armazenam os dados em pares chave-valor:

- Os dados são acessados por uma chave (de busca):



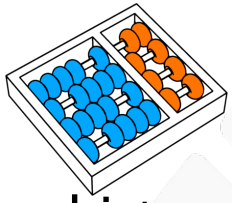
Dicionários

Listas armazenam os dados de acordo com os índices:

- Os dados são acessados pelo índice:
  - Um **int**.
- Para cada índice, temos um valor:
  - É uma função no sentido matemático.
- Podemos pensar que os nossos dados são pares índice-valor.

Dicionários armazenam os dados em pares chave-valor:

- Os dados são acessados por uma chave (de busca):
  - Pode ser **str**, **int**, **float**.



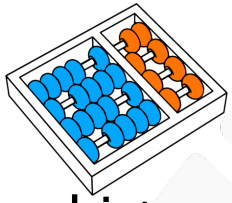
Dicionários

Listas armazenam os dados de acordo com os índices:

- Os dados são acessados pelo índice:
  - Um **int**.
- Para cada índice, temos um valor:
  - É uma função no sentido matemático.
- Podemos pensar que os nossos dados são pares índice-valor.

Dicionários armazenam os dados em pares chave-valor:

- Os dados são acessados por uma chave (de busca):
  - Pode ser **str**, **int**, **float**.
  - Até outros objetos (mas não qualquer objeto).



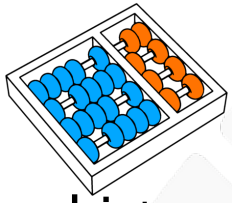
Dicionários

Listas armazenam os dados de acordo com os índices:

- Os dados são acessados pelo índice:
  - Um **int**.
- Para cada índice, temos um valor:
  - É uma função no sentido matemático.
- Podemos pensar que os nossos dados são pares índice-valor.

Dicionários armazenam os dados em pares chave-valor:

- Os dados são acessados por uma chave (de busca):
  - Pode ser **str**, **int**, **float**.
  - Até outros objetos (mas não qualquer objeto).
- Para cada chave, temos um valor:



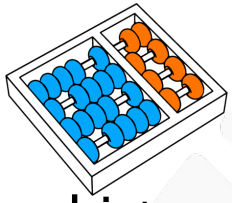
Dicionários

Listas armazenam os dados de acordo com os índices:

- Os dados são acessados pelo índice:
  - Um **int**.
- Para cada índice, temos um valor:
  - É uma função no sentido matemático.
- Podemos pensar que os nossos dados são pares índice-valor.

Dicionários armazenam os dados em pares chave-valor:

- Os dados são acessados por uma chave (de busca):
  - Pode ser **str**, **int**, **float**.
  - Até outros objetos (mas não qualquer objeto).
- Para cada chave, temos um valor:
  - Também é uma função no sentido matemático.



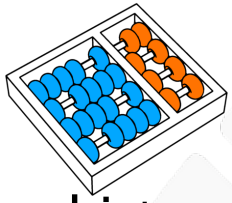
Dicionários

Listas armazenam os dados de acordo com os índices:

- Os dados são acessados pelo índice:
  - Um **int**.
- Para cada índice, temos um valor:
  - É uma função no sentido matemático.
- Podemos pensar que os nossos dados são pares índice-valor.

Dicionários armazenam os dados em pares chave-valor:

- Os dados são acessados por uma chave (de busca):
  - Pode ser **str**, **int**, **float**.
  - Até outros objetos (mas não qualquer objeto).
- Para cada chave, temos um valor:
  - Também é uma função no sentido matemático.
- Exemplos:



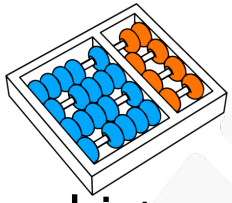
Dicionários

Listas armazenam os dados de acordo com os índices:

- Os dados são acessados pelo índice:
  - Um **int**.
- Para cada índice, temos um valor:
  - É uma função no sentido matemático.
- Podemos pensar que os nossos dados são pares índice-valor.

Dicionários armazenam os dados em pares chave-valor:

- Os dados são acessados por uma chave (de busca):
  - Pode ser **str**, **int**, **float**.
  - Até outros objetos (mas não qualquer objeto).
- Para cada chave, temos um valor:
  - Também é uma função no sentido matemático.
- Exemplos:
  - **ra["ana"] = 123456**.



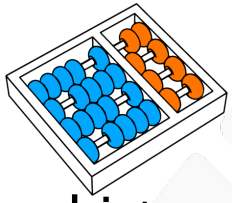
Dicionários

Listas armazenam os dados de acordo com os índices:

- Os dados são acessados pelo índice:
  - Um **int**.
- Para cada índice, temos um valor:
  - É uma função no sentido matemático.
- Podemos pensar que os nossos dados são pares índice-valor.

Dicionários armazenam os dados em pares chave-valor:

- Os dados são acessados por uma chave (de busca):
  - Pode ser **str**, **int**, **float**.
  - Até outros objetos (mas não qualquer objeto).
- Para cada chave, temos um valor:
  - Também é uma função no sentido matemático.
- Exemplos:
  - **ra["ana"] = 123456**.
  - **nome[123456] = "ana"**.



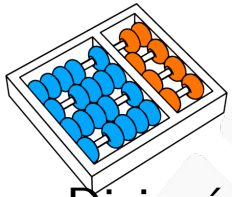
Dicionários

Listas armazenam os dados de acordo com os índices:

- Os dados são acessados pelo índice:
  - Um **int**.
- Para cada índice, temos um valor:
  - É uma função no sentido matemático.
- Podemos pensar que os nossos dados são pares índice-valor.

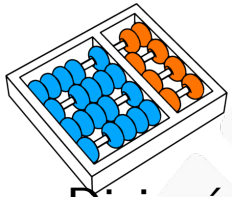
Dicionários armazenam os dados em pares chave-valor:

- Os dados são acessados por uma chave (de busca):
  - Pode ser **str**, **int**, **float**.
  - Até outros objetos (mas não qualquer objeto).
- Para cada chave, temos um valor:
  - Também é uma função no sentido matemático.
- Exemplos:
  - **ra["ana"] = 123456**.
  - **nome[123456] = "ana"**.
  - **d[3.2] = 10**.



Criando um Dicionário

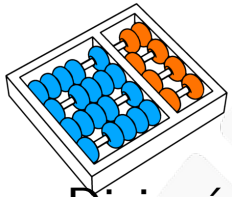
Dicionário vazio:



Criando um Dicionário

Dicionário vazio:

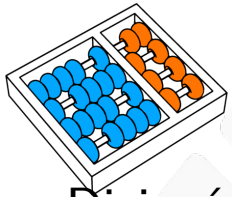
- `d = { }` (assim como `l = [ ]`)



Criando um Dicionário

Dicionário vazio:

- **d = { }** (assim como **l = [ ]**)
- **d = dict()** (assim como **l = list()**)

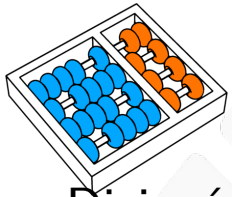


Criando um Dicionário

Dicionário vazio:

- **d = { }** (assim como **l = [ ]**)
- **d = dict()** (assim como **l = list()**)

Dicionário com conteúdo inicial:



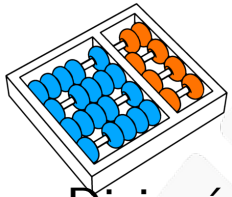
Criando um Dicionário

Dicionário vazio:

- **d = { }** (assim como **l = [ ]**)
- **d = dict()** (assim como **l = list()**)

Dicionário com conteúdo inicial:

- Podemos passar os pares no formato **chave: valor**.



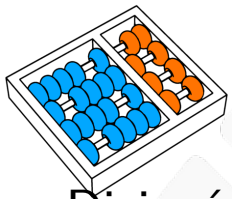
Criando um Dicionário

Dicionário vazio:

- **d = { }** (assim como **l = [ ]**)
- **d = dict()** (assim como **l = list()**)

Dicionário com conteúdo inicial:

- Podemos passar os pares no formato **chave: valor**.
  - **d = {"ana": 123456}**.



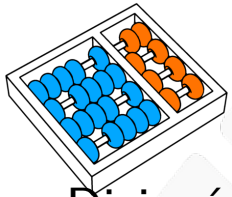
Criando um Dicionário

Dicionário vazio:

- **d = { }** (assim como **l = [ ]**)
- **d = dict()** (assim como **l = list()**)

Dicionário com conteúdo inicial:

- Podemos passar os pares no formato **chave: valor**.
  - **d = {"ana": 123456}**.
  - **d = {"ana": 123456, "beto": 123457}**.



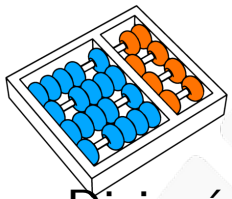
Criando um Dicionário

Dicionário vazio:

- **d = { }** (assim como **l = [ ]**)
- **d = dict()** (assim como **l = list()**)

Dicionário com conteúdo inicial:

- Podemos passar os pares no formato **chave: valor**.
  - **d = {"ana": 123456}**.
  - **d = {"ana": 123456, "beto": 123457}**.
- Ou usando o construtor **dict**:



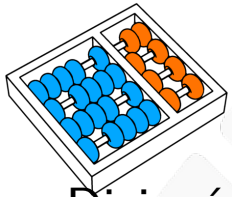
Criando um Dicionário

Dicionário vazio:

- **d = { }** (assim como **l = [ ]**)
- **d = dict()** (assim como **l = list()**)

Dicionário com conteúdo inicial:

- Podemos passar os pares no formato **chave: valor**.
  - **d = {"ana": 123456}**.
  - **d = {"ana": 123456, "beto": 123457}**.
- Ou usando o construtor **dict**:
  - **d = dict([["ana", 123456], ["beto", 123457]])**.



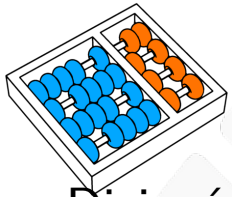
Criando um Dicionário

Dicionário vazio:

- **d = { }** (assim como **l = [ ]**)
- **d = dict()** (assim como **l = list()**)

Dicionário com conteúdo inicial:

- Podemos passar os pares no formato **chave: valor**.
  - **d = {"ana": 123456}**.
  - **d = {"ana": 123456, "beto": 123457}**.
- Ou usando o construtor **dict**:
  - **d = dict([["ana", 123456], ["beto", 123457]])**.
- Se **todas** as chaves forem strings, podemos fazer:



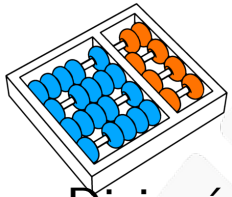
Criando um Dicionário

Dicionário vazio:

- **d = { }** (assim como **l = [ ]**)
- **d = dict()** (assim como **l = list()**)

Dicionário com conteúdo inicial:

- Podemos passar os pares no formato **chave: valor**.
  - **d = {"ana": 123456}**.
  - **d = {"ana": 123456, "beto": 123457}**.
- Ou usando o construtor **dict**:
  - **d = dict([["ana", 123456], ["beto", 123457]])**.
- Se **todas** as chaves forem strings, podemos fazer:
  - **d = dict(ana=123456, beto=123457)**.



Criando um Dicionário

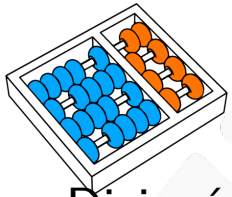
Dicionário vazio:

- **d = { }** (assim como **l = [ ]**)
- **d = dict()** (assim como **l = list()**)

Dicionário com conteúdo inicial:

- Podemos passar os pares no formato **chave: valor**.
  - **d = {"ana": 123456}**.
  - **d = {"ana": 123456, "beto": 123457}**.
- Ou usando o construtor **dict**:
  - **d = dict([["ana", 123456], ["beto", 123457]])**.
- Se **todas** as chaves forem strings, podemos fazer:
  - **d = dict(ana=123456, beto=123457)**.

As chaves de um dicionário podem ser de tipos diferentes:



Criando um Dicionário

Dicionário vazio:

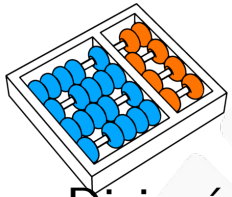
- **d = { }** (assim como **l = [ ]**)
- **d = dict()** (assim como **l = list()**)

Dicionário com conteúdo inicial:

- Podemos passar os pares no formato **chave: valor**.
  - **d = {"ana": 123456}**.
  - **d = {"ana": 123456, "beto": 123457}**.
- Ou usando o construtor **dict**:
  - **d = dict([["ana", 123456], ["beto", 123457]])**.
- Se **todas** as chaves forem strings, podemos fazer:
  - **d = dict(ana=123456, beto=123457)**.

As chaves de um dicionário podem ser de tipos diferentes:

- Mas não temos chaves repetidas.



Criando um Dicionário

Dicionário vazio:

- **d = { }** (assim como **l = [ ]**)
- **d = dict()** (assim como **l = list()**)

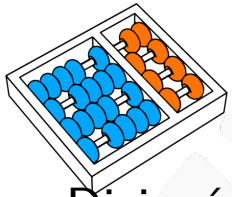
Dicionário com conteúdo inicial:

- Podemos passar os pares no formato **chave: valor**.
  - **d = {"ana": 123456}**.
  - **d = {"ana": 123456, "beto": 123457}**.
- Ou usando o construtor **dict**:
  - **d = dict([["ana", 123456], ["beto", 123457]])**.
- Se **todas** as chaves forem strings, podemos fazer:
  - **d = dict(ana=123456, beto=123457)**.

As chaves de um dicionário podem ser de tipos diferentes:

- Mas não temos chaves repetidas.

Os valores também podem ser de tipos diferentes:



Criando um Dicionário

Dicionário vazio:

- **d = { }** (assim como **l = [ ]**)
- **d = dict()** (assim como **l = list()**)

Dicionário com conteúdo inicial:

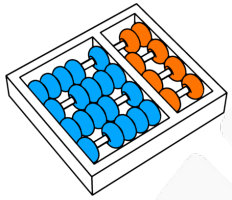
- Podemos passar os pares no formato **chave: valor**.
  - **d = {"ana": 123456}**.
  - **d = {"ana": 123456, "beto": 123457}**.
- Ou usando o construtor **dict**:
  - **d = dict([["ana", 123456], ["beto", 123457]])**.
- Se **todas** as chaves forem strings, podemos fazer:
  - **d = dict(ana=123456, beto=123457)**.

As chaves de um dicionário podem ser de tipos diferentes:

- Mas não temos chaves repetidas.

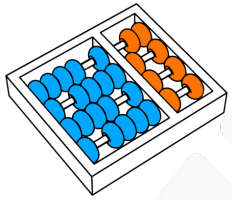
Os valores também podem ser de tipos diferentes:

- E podem ser repetidos.



Leitura e Escrita

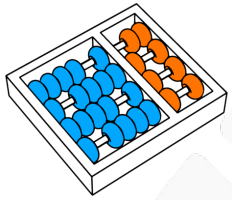
Para ler um valor a partir de uma chave, escrevemos:



Leitura e Escrita

Para ler um valor a partir de uma chave, escrevemos:

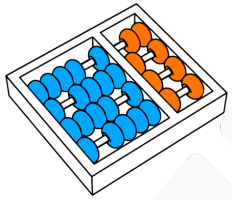
- **dicionario[chave]**.



Leitura e Escrita

Para ler um valor a partir de uma chave, escrevemos:

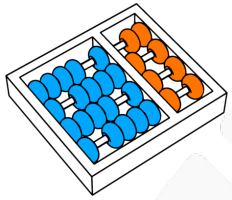
- **dicionario[chave]**.
- Ex: **print(ra["ana"])**.



Leitura e Escrita

Para ler um valor a partir de uma chave, escrevemos:

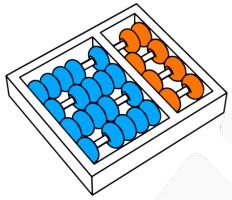
- **dicionario[chave]**.
- Ex: **print(ra["ana"])**.
- Recebemos um **KeyError** se a chave não existe.



Leitura e Escrita

Para ler um valor a partir de uma chave, escrevemos:

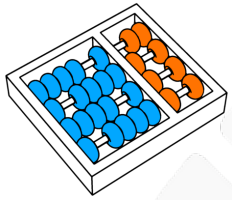
- **dicionario[chave]**.
- Ex: **print(ra["ana"])**.
- Recebemos um **KeyError** se a chave não existe.
- Podemos testar se a chave existe: **chave in dicionario**:



Leitura e Escrita

Para ler um valor a partir de uma chave, escrevemos:

- **dicionario[chave]**.
- Ex: **print(ra["ana"])**.
- Recebemos um **KeyError** se a chave não existe.
- Podemos testar se a chave existe: **chave in dicionario**:
  - Ex: **"ana" in ra**.

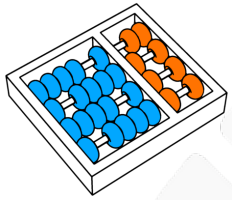


Leitura e Escrita

Para ler um valor a partir de uma chave, escrevemos:

- **dicionario[chave]**.
- Ex: **print(ra["ana"])**.
- Recebemos um **KeyError** se a chave não existe.
- Podemos testar se a chave existe: **chave in dicionario**:
  - Ex: **"ana" in ra**.

Para escrever um valor a partir de uma chave, escrevemos:



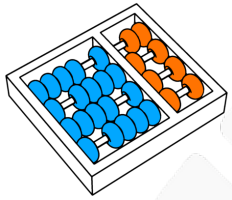
Leitura e Escrita

Para ler um valor a partir de uma chave, escrevemos:

- **dicionario[chave]**.
- Ex: **print(ra["ana"])**.
- Recebemos um **KeyError** se a chave não existe.
- Podemos testar se a chave existe: **chave in dicionario**:
  - Ex: **"ana" in ra**.

Para escrever um valor a partir de uma chave, escrevemos:

- **dicionario[chave] = valor**.



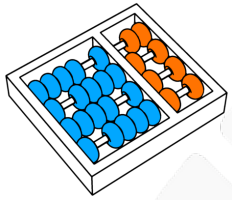
Leitura e Escrita

Para ler um valor a partir de uma chave, escrevemos:

- **dicionario[chave]**.
- Ex: **print(ra["ana"])**.
- Recebemos um **KeyError** se a chave não existe.
- Podemos testar se a chave existe: **chave in dicionario**:
  - Ex: **"ana" in ra**.

Para escrever um valor a partir de uma chave, escrevemos:

- **dicionario[chave] = valor**.
- **ra["ana"] = 123456**.



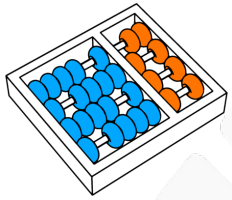
Leitura e Escrita

Para ler um valor a partir de uma chave, escrevemos:

- **dicionario[chave]**.
- Ex: **print(ra["ana"])**.
- Recebemos um **KeyError** se a chave não existe.
- Podemos testar se a chave existe: **chave in dicionario**:
  - Ex: **"ana" in ra**.

Para escrever um valor a partir de uma chave, escrevemos:

- **dicionario[chave] = valor**.
- **ra["ana"] = 123456**.
- Podemos fazer isso mesmo se a chave não existir!



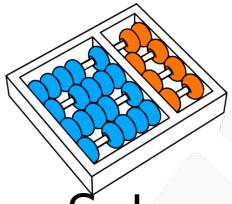
Leitura e Escrita

Para ler um valor a partir de uma chave, escrevemos:

- **dicionario[chave]**.
- Ex: **print(ra["ana"])**.
- Recebemos um **KeyError** se a chave não existe.
- Podemos testar se a chave existe: **chave in dicionario**:
  - Ex: **"ana" in ra**.

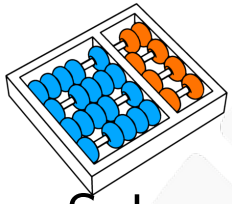
Para escrever um valor a partir de uma chave, escrevemos:

- **dicionario[chave] = valor**.
- **ra["ana"] = 123456**.
- Podemos fazer isso mesmo se a chave não existir!
  - Essa é uma forma de adicionar pares no dicionário.



Iterando

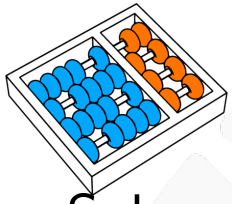
Sobre as chaves:



Iterando

Sobre as chaves:

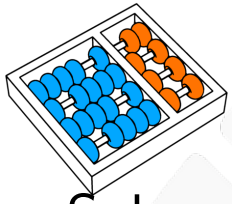
- Basta usar **for chave in d:**



Iterando

Sobre as chaves:

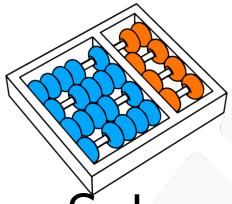
- Basta usar **for chave in d:**
- Ou **for chave in d.keys():**



Iterando

Sobre as chaves:

- Basta usar **for chave in d:**
- Ou **for chave in d.keys():**
- Lembre-se que não há chave repetida.

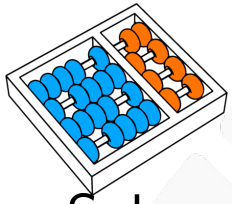


Iterando

Sobre as chaves:

- Basta usar **for chave in d:**
- Ou **for chave in d.keys():**
- Lembre-se que não há chave repetida.

Sobre os valores:



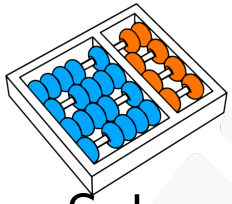
Iterando

Sobre as chaves:

- Basta usar **for chave in d:**
- Ou **for chave in d.keys():**
- Lembre-se que não há chave repetida.

Sobre os valores:

- **for valor in d.values():**



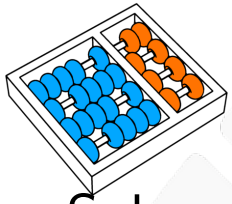
Iterando

Sobre as chaves:

- Basta usar **for chave in d:**
- Ou **for chave in d.keys():**
- Lembre-se que não há chave repetida.

Sobre os valores:

- **for valor in d.values():**
- Lembre-se que pode haver valor repetido.



Iterando

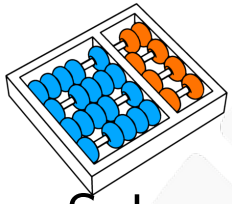
Sobre as chaves:

- Basta usar **for chave in d:**
- Ou **for chave in d.keys():**
- Lembre-se que não há chave repetida.

Sobre os valores:

- **for valor in d.values():**
- Lembre-se que pode haver valor repetido.

Sobre os pares:



Iterando

Sobre as chaves:

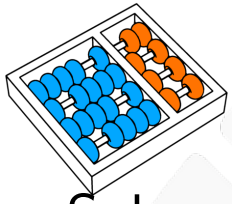
- Basta usar **for chave in d:**
- Ou **for chave in d.keys():**
- Lembre-se que não há chave repetida.

Sobre os valores:

- **for valor in d.values():**
- Lembre-se que pode haver valor repetido.

Sobre os pares:

- **for chave, valor in d.items():**



Iterando

Sobre as chaves:

- Basta usar **for chave in d:**
- Ou **for chave in d.keys():**
- Lembre-se que não há chave repetida.

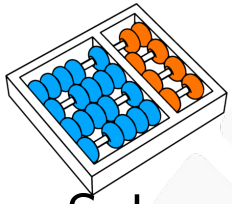
Sobre os valores:

- **for valor in d.values():**
- Lembre-se que pode haver valor repetido.

Sobre os pares:

- **for chave, valor in d.items():**

Importante: No Python 3.7 em diante a ordem de acesso das chaves é a ordem de inserção no dicionário:



Iterando

Sobre as chaves:

- Basta usar **for chave in d:**
- Ou **for chave in d.keys():**
- Lembre-se que não há chave repetida.

Sobre os valores:

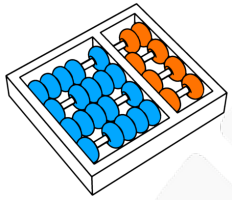
- **for valor in d.values():**
- Lembre-se que pode haver valor repetido.

Sobre os pares:

- **for chave, valor in d.items():**

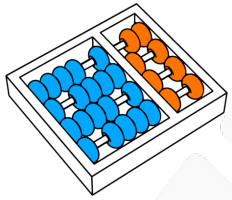
Importante: No Python 3.7 em diante a ordem de acesso das chaves é a ordem de inserção no dicionário:

- Não é necessariamente verdade em versões anteriores.



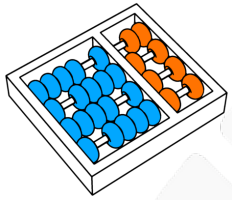
Outras informações

- Você pode apagar escrevendo **del dicionario[chave]**.



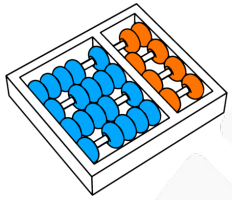
Outras informações

- Você pode apagar escrevendo **del dicionario[chave]**.
- Você pode saber quantos pares há escrevendo **len(dicionario)**.



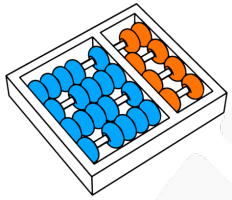
Outras informações

- Você pode apagar escrevendo **del dicionario[chave]**.
- Você pode saber quantos pares há escrevendo **len(dicionario)**.
- Você pode usar o método **get** ao invés das chaves para evitar **KeyError**:



Outras informações

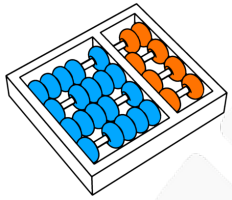
- Você pode apagar escrevendo **del dicionario[chave]**.
- Você pode saber quantos pares há escrevendo **len(dicionario)**.
- Você pode usar o método **get** ao invés das chaves para evitar **KeyError**:
  - **d.get(chave, alt)**: se **chave** existir, devolve o item correspondente, senão devolve **alt**.



Outras informações

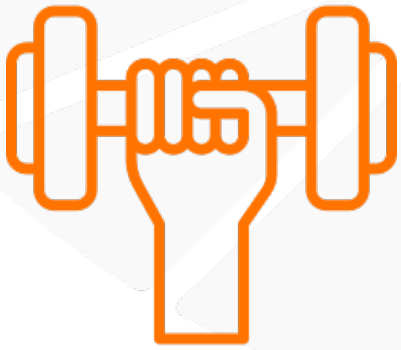
- Você pode apagar escrevendo **del dicionario[chave]**.
- Você pode saber quantos pares há escrevendo **len(dicionario)**.
- Você pode usar o método **get** ao invés das chaves para evitar **KeyError**:
  - **d.get(chave, alt)**: se **chave** existir, devolve o item correspondente, senão devolve **alt**.
- Há outros métodos também, veja a documentação!

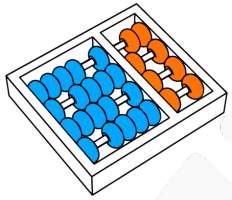
Exercícios



Dicionários

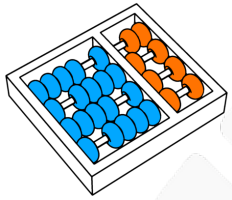
Vamos fazer alguns exercícios?





Exercícios

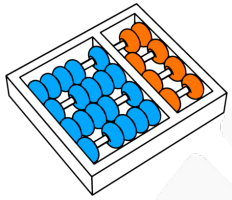
1. Faça um programa que permita buscar o RA de um aluno a partir do seu nome.



Exercícios

1. Faça um programa que permita buscar o RA de um aluno a partir do seu nome.

2. Faça um programa, que dado uma lista de número inteiros em $[0, 10)$, imprime um histograma para tais números.



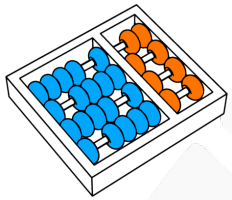
Exercícios

1. Faça um programa que permita buscar o RA de um aluno a partir do seu nome.

2. Faça um programa, que dado uma lista de número inteiros em $[0, 10)$, imprime um histograma para tais números.

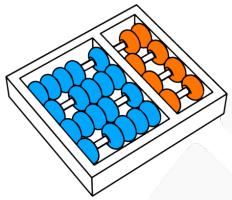
3. Repita o exercício anterior utilizando nomes de alunos.

Conjuntos



O que são?

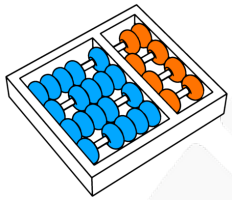
Conjuntos lembram dicionários:



O que são?

Conjuntos lembram dicionários:

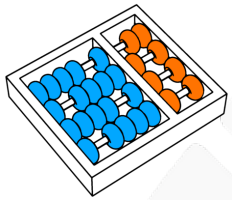
- Porém, ao invés de armazenar pares chave-valor.



O que são?

Conjuntos lembram dicionários:

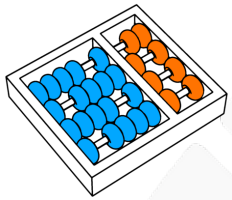
- Porém, ao invés de armazenar pares chave-valor.
- Armazenam apenas chaves.



O que são?

Conjuntos lembram dicionários:

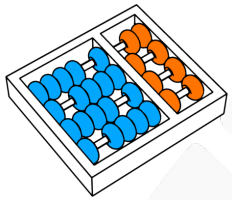
- Porém, ao invés de armazenar pares chave-valor.
- Armazenam apenas chaves.
- As chaves podem ser **int**, **str**, **float**.



O que são?

Conjuntos lembram dicionários:

- Porém, ao invés de armazenar pares chave-valor.
- Armazenam apenas chaves.
- As chaves podem ser **int**, **str**, **float**.
- E até outros objetos... mas não todos...

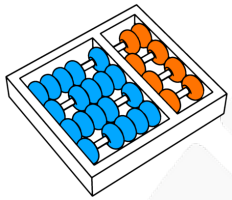


O que são?

Conjuntos lembram dicionários:

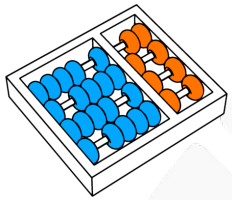
- Porém, ao invés de armazenar pares chave-valor.
- Armazenam apenas chaves.
- As chaves podem ser **int**, **str**, **float**.
- E até outros objetos... mas não todos...

Como na matemática, conjuntos não têm repetição!



Criando conjuntos

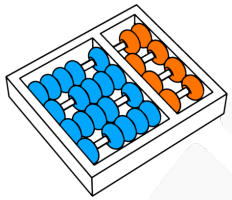
Conjuntos com valores iniciais:



Criando conjuntos

Conjuntos com valores iniciais:

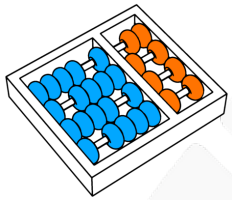
- **s = {"ana", "beto", "carlos"}**.



Criando conjuntos

Conjuntos com valores iniciais:

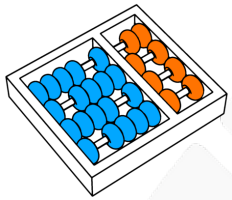
- **s = {"ana", "beto", "carlos"}**.
  - Notação similar a do dicionário.



Criando conjuntos

Conjuntos com valores iniciais:

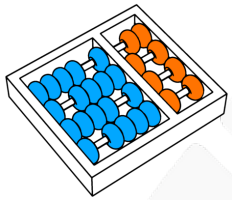
- **s = {"ana", "beto", "carlos"}**.
  - Notação similar a do dicionário.
  - Mas sem os valores, apenas chaves.



Criando conjuntos

Conjuntos com valores iniciais:

- **s = {"ana", "beto", "carlos"}**.
  - Notação similar a do dicionário.
  - Mas sem os valores, apenas chaves.
- **s = set(["ana", "beto", "carlos"])**.

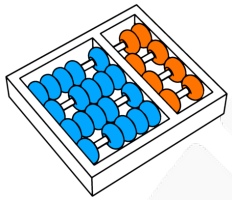


Criando conjuntos

Conjuntos com valores iniciais:

- **s = {"ana", "beto", "carlos"}**.
  - Notação similar a do dicionário.
  - Mas sem os valores, apenas chaves.
- **s = set(["ana", "beto", "carlos"])**.

Conjuntos vazios:



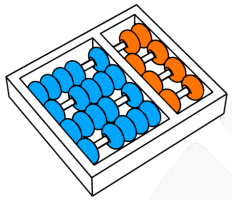
Criando conjuntos

Conjuntos com valores iniciais:

- **s = {"ana", "beto", "carlos"}**.
  - Notação similar a do dicionário.
  - Mas sem os valores, apenas chaves.
- **s = set(["ana", "beto", "carlos"])**.

Conjuntos vazios:

- **s = set()**.



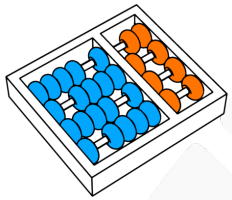
Criando conjuntos

Conjuntos com valores iniciais:

- **s = {"ana", "beto", "carlos"}**.
  - Notação similar a do dicionário.
  - Mas sem os valores, apenas chaves.
- **s = set(["ana", "beto", "carlos"])**.

Conjuntos vazios:

- **s = set()**.
- Não podemos escrever **s = {}**.



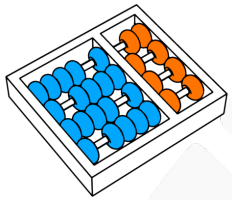
Criando conjuntos

Conjuntos com valores iniciais:

- **s = {"ana", "beto", "carlos"}**.
  - Notação similar a do dicionário.
  - Mas sem os valores, apenas chaves.
- **s = set(["ana", "beto", "carlos"])**.

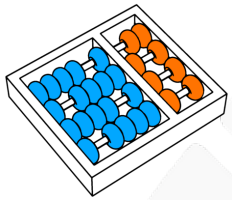
Conjuntos vazios:

- **s = set()**.
- Não podemos escrever **s = {}**.
  - Porque isso cria um dicionário vazio!



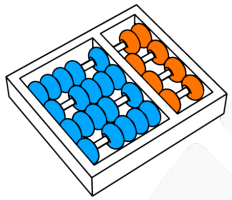
Manipulando conjuntos

- **s.add(x)** adiciona **x**:



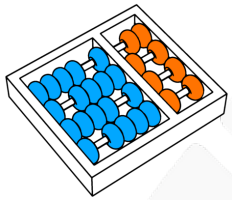
Manipulando conjuntos

- **s.add(x)** adiciona **x**:
  - Apenas se **x** não estiver no conjunto.



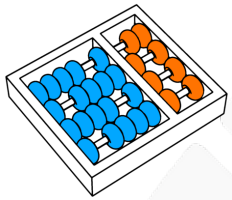
Manipulando conjuntos

- **s.add(x)** adiciona **x**:
  - Apenas se **x** não estiver no conjunto.
- **del** não funciona. . . (conjuntos não tem pares chave-valor).



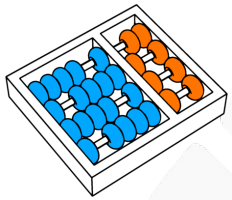
Manipulando conjuntos

- **s.add(x)** adiciona **x**:
  - Apenas se **x** não estiver no conjunto.
- **del** não funciona. . . (conjuntos não tem pares chave-valor).
- **s.discard(x)** remove **x**:



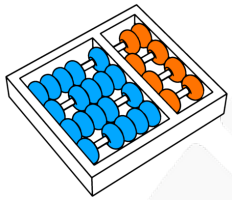
Manipulando conjuntos

- **s.add(x)** adiciona **x**:
  - Apenas se **x** não estiver no conjunto.
- **del** não funciona. . . (conjuntos não tem pares chave-valor).
- **s.discard(x)** remove **x**:
  - Se **x** não está em **s**, nada acontece.



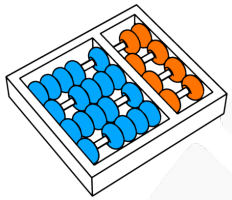
Manipulando conjuntos

- **s.add(x)** adiciona **x**:
  - Apenas se **x** não estiver no conjunto.
- **del** não funciona. . . (conjuntos não tem pares chave-valor).
- **s.discard(x)** remove **x**:
  - Se **x** não está em **s**, nada acontece.
- **s.remove(x)** remove **x**:



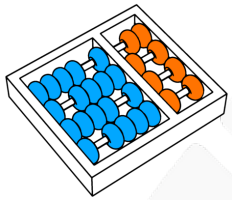
Manipulando conjuntos

- **s.add(x)** adiciona **x**:
  - Apenas se **x** não estiver no conjunto.
- **del** não funciona. . . (conjuntos não tem pares chave-valor).
- **s.discard(x)** remove **x**:
  - Se **x** não está em **s**, nada acontece.
- **s.remove(x)** remove **x**:
  - Se **x** não está em **s**, temos um **KeyError**.



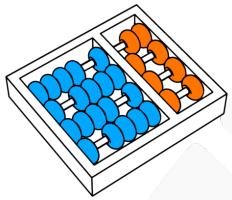
Manipulando conjuntos

- **s.add(x)** adiciona **x**:
  - Apenas se **x** não estiver no conjunto.
- **del** não funciona. . . (conjuntos não tem pares chave-valor).
- **s.discard(x)** remove **x**:
  - Se **x** não está em **s**, nada acontece.
- **s.remove(x)** remove **x**:
  - Se **x** não está em **s**, temos um **KeyError**.
- **x in s** para verificar se **x** está em **s**.



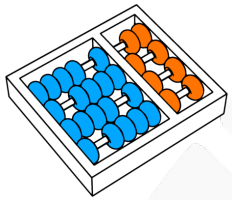
Manipulando conjuntos

- **s.add(x)** adiciona **x**:
  - Apenas se **x** não estiver no conjunto.
- **del** não funciona. . . (conjuntos não tem pares chave-valor).
- **s.discard(x)** remove **x**:
  - Se **x** não está em **s**, nada acontece.
- **s.remove(x)** remove **x**:
  - Se **x** não está em **s**, temos um **KeyError**.
- **x in s** para verificar se **x** está em **s**.
- **for x in s:** para iterar sobre **s**:



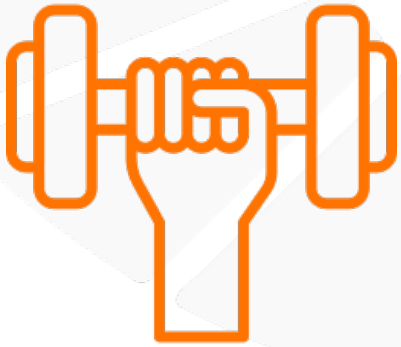
Manipulando conjuntos

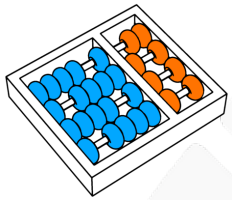
- **s.add(x)** adiciona **x**:
  - Apenas se **x** não estiver no conjunto.
- **del** não funciona. . . (conjuntos não tem pares chave-valor).
- **s.discard(x)** remove **x**:
  - Se **x** não está em **s**, nada acontece.
- **s.remove(x)** remove **x**:
  - Se **x** não está em **s**, temos um **KeyError**.
- **x in s** para verificar se **x** está em **s**.
- **for x in s:** para iterar sobre **s**:
  - A ordem de acesso não é necessariamente a ordem de inserção no conjunto!



Conjuntos

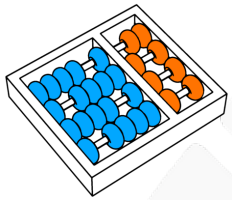
Vamos fazer alguns exercícios?





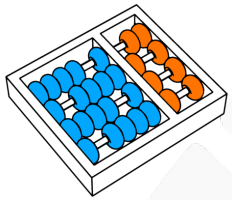
Exercícios

1. Faça uma função que, dados conjuntos **s1** e **s2**, devolve um novo conjunto representando a união de **s1** e **s2**.



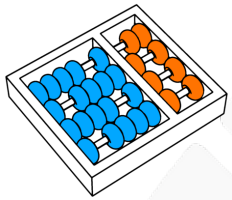
Exercícios

1. Faça uma função que, dados conjuntos **s1** e **s2**, devolve um novo conjunto representando a união de **s1** e **s2**.
2. Faça uma função que, dados conjuntos **s1** e **s2**, devolve um novo conjunto representando a interseção de **s1** e **s2**.



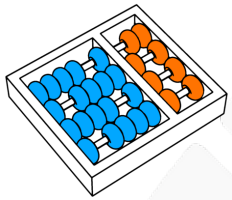
Exercícios

1. Faça uma função que, dados conjuntos **s1** e **s2**, devolve um novo conjunto representando a união de **s1** e **s2**.
2. Faça uma função que, dados conjuntos **s1** e **s2**, devolve um novo conjunto representando a interseção de **s1** e **s2**.
3. Faça uma função que, dados conjuntos **s1** e **s2**, devolve um novo conjunto representando a subtração de **s2** em **s1**.



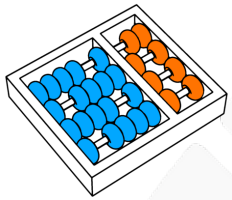
Exercícios

1. Faça uma função que, dados conjuntos **s1** e **s2**, devolve um novo conjunto representando a união de **s1** e **s2**.
2. Faça uma função que, dados conjuntos **s1** e **s2**, devolve um novo conjunto representando a interseção de **s1** e **s2**.
3. Faça uma função que, dados conjuntos **s1** e **s2**, devolve um novo conjunto representando a subtração de **s2** em **s1**.
4. Faça uma função que, dados conjuntos **s1** e **s2**, devolve se **s1** é subconjunto de **s2**.



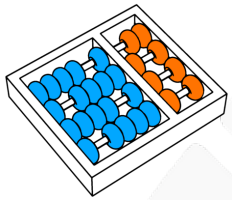
Exercícios

1. Faça uma função que, dados conjuntos **s1** e **s2**, devolve um novo conjunto representando a união de **s1** e **s2**.
2. Faça uma função que, dados conjuntos **s1** e **s2**, devolve um novo conjunto representando a interseção de **s1** e **s2**.
3. Faça uma função que, dados conjuntos **s1** e **s2**, devolve um novo conjunto representando a subtração de **s2** em **s1**.
4. Faça uma função que, dados conjuntos **s1** e **s2**, devolve se **s1** é subconjunto de **s2**.
5. Faça uma função que, dada uma lista, devolve uma nova lista sem elementos repetidos. Dica: use conjuntos!



Métodos úteis de conjunto

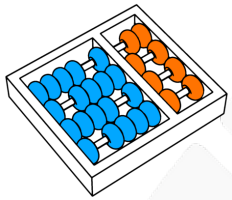
Devolvem um novo conjunto:



Métodos úteis de conjunto

Devolvem um novo conjunto:

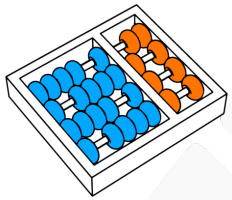
- **s1.union(s2)**: união.



Métodos úteis de conjunto

Devolvem um novo conjunto:

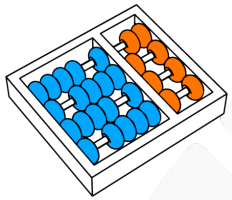
- **s1.union(s2)**: união.
  - Ou **s1 | s2**.



Métodos úteis de conjunto

Devolvem um novo conjunto:

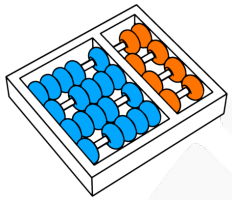
- **s1.union(s2)**: união.
  - Ou **s1** | **s2**.
  - O símbolo | representa **ou**.



Métodos úteis de conjunto

Devolvem um novo conjunto:

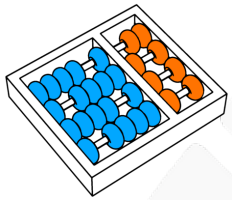
- **s1.union(s2)**: união.
  - Ou **s1 | s2**.
  - O símbolo | representa **ou**.
- **s1.intersection(s2)**: interseção.



Métodos úteis de conjunto

Devolvem um novo conjunto:

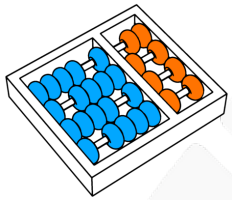
- **s1.union(s2)**: união.
  - Ou **s1 | s2**.
  - O símbolo | representa **ou**.
- **s1.intersection(s2)**: interseção.
  - Ou **s1 & s2**.



Métodos úteis de conjunto

Devolvem um novo conjunto:

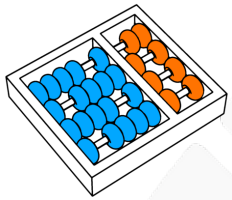
- **s1.union(s2)**: união.
  - Ou **s1 | s2**.
  - O símbolo | representa **ou**.
- **s1.intersection(s2)**: interseção.
  - Ou **s1 & s2**.
  - O símbolo **&** representa **e**.



Métodos úteis de conjunto

Devolvem um novo conjunto:

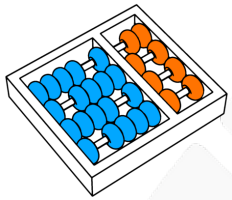
- **s1.union(s2)**: união.
  - Ou **s1 | s2**.
  - O símbolo | representa **ou**.
- **s1.intersection(s2)**: interseção.
  - Ou **s1 & s2**.
  - O símbolo **&** representa **e**.
- **s1.difference(s2, ...)**: diferença.



Métodos úteis de conjunto

Devolvem um novo conjunto:

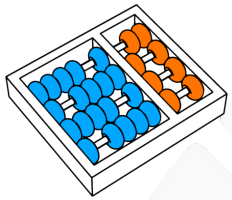
- **s1.union(s2)**: união.
  - Ou **s1 | s2**.
  - O símbolo | representa **ou**.
- **s1.intersection(s2)**: interseção.
  - Ou **s1 & s2**.
  - O símbolo **&** representa **e**.
- **s1.difference(s2, ...)**: diferença.
  - Ou **s1 - s2**.



Métodos úteis de conjunto

Devolvem um novo conjunto:

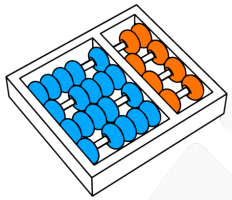
- **s1.union(s2)**: união.
  - Ou **s1 | s2**.
  - O símbolo | representa **ou**.
- **s1.intersection(s2)**: interseção.
  - Ou **s1 & s2**.
  - O símbolo **&** representa **e**.
- **s1.difference(s2, ...)**: diferença.
  - Ou **s1 - s2**.
- **s1.symmetric_difference(s2)**: diferença simétrica.



Métodos úteis de conjunto

Devolvem um novo conjunto:

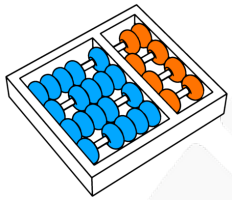
- **s1.union(s2)**: união.
  - Ou **s1 | s2**.
  - O símbolo | representa **ou**.
- **s1.intersection(s2)**: interseção.
  - Ou **s1 & s2**.
  - O símbolo **&** representa **e**.
- **s1.difference(s2, ...)**: diferença.
  - Ou **s1 - s2**.
- **s1.symmetric_difference(s2)**: diferença simétrica.
  - Ou **s1 ^ s2**.



Métodos úteis de conjunto

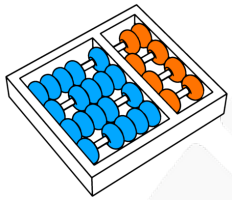
Devolvem um novo conjunto:

- **s1.union(s2)**: união.
  - Ou **s1 | s2**.
  - O símbolo | representa **ou**.
- **s1.intersection(s2)**: interseção.
  - Ou **s1 & s2**.
  - O símbolo **&** representa **e**.
- **s1.difference(s2, ...)**: diferença.
  - Ou **s1 - s2**.
- **s1.symmetric_difference(s2)**: diferença simétrica.
  - Ou **s1 ^ s2**.
  - O símbolo ^ representa **ou exclusivo**.



Métodos úteis de conjunto

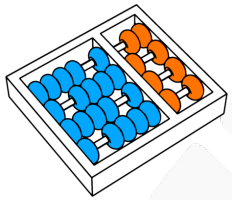
Alteram o próprio conjunto:



Métodos úteis de conjunto

Alteram o próprio conjunto:

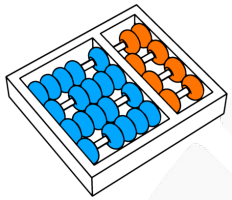
- **s1.update(s2)**: altera **s1** para a união de **s1** e **s2**.



Métodos úteis de conjunto

Alteram o próprio conjunto:

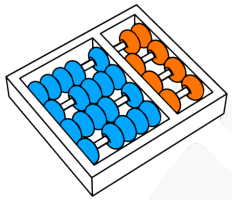
- **s1.update(s2)**: altera **s1** para a união de **s1** e **s2**.
  - Ou **s1 |= s2**.



Métodos úteis de conjunto

Alteram o próprio conjunto:

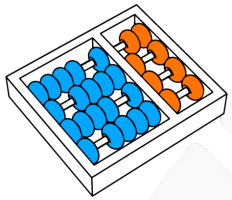
- **s1.update(s2)**: altera **s1** para a união de **s1** e **s2**.
  - Ou **s1 |= s2**.
- **s1.intersection_update(s2)**: interseção.



Métodos úteis de conjunto

Alteram o próprio conjunto:

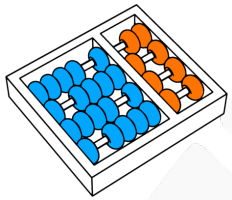
- **s1.update(s2)**: altera **s1** para a união de **s1** e **s2**.
  - Ou **s1 |= s2**.
- **s1.intersection_update(s2)**: interseção.
  - Ou **s1 &= s2**.



Métodos úteis de conjunto

Alteram o próprio conjunto:

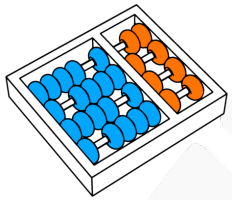
- **s1.update(s2)**: altera **s1** para a união de **s1** e **s2**.
  - Ou **s1 |= s2**.
- **s1.intersection_update(s2)**: interseção.
  - Ou **s1 &= s2**.
- **s1.difference_update(s2, ...)**: diferença.



Métodos úteis de conjunto

Alteram o próprio conjunto:

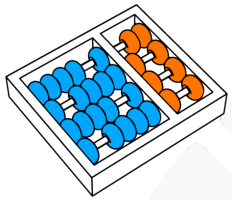
- **s1.update(s2)**: altera **s1** para a união de **s1** e **s2**.
  - Ou **s1 |= s2**.
- **s1.intersection_update(s2)**: interseção.
  - Ou **s1 &= s2**.
- **s1.difference_update(s2, ...)**: diferença.
  - Ou **s1 -= s2**.



Métodos úteis de conjunto

Alteram o próprio conjunto:

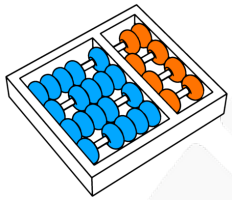
- **s1.update(s2)**: altera **s1** para a união de **s1** e **s2**.
  - Ou **s1 |= s2**.
- **s1.intersection_update(s2)**: interseção.
  - Ou **s1 &= s2**.
- **s1.difference_update(s2, ...)**: diferença.
  - Ou **s1 -= s2**.
- **s1.symmetric_difference_update(s2)**: diferença simétrica.



Métodos úteis de conjunto

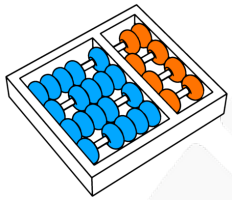
Alteram o próprio conjunto:

- **s1.update(s2)**: altera **s1** para a união de **s1** e **s2**.
  - Ou **s1 |= s2**.
- **s1.intersection_update(s2)**: interseção.
  - Ou **s1 &= s2**.
- **s1.difference_update(s2, ...)**: diferença.
  - Ou **s1 -= s2**.
- **s1.symmetric_difference_update(s2)**: diferença simétrica.
  - Ou **s1 ^= s2**.



Métodos úteis de conjunto

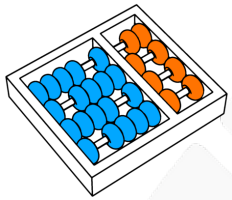
Relações:



Métodos úteis de conjunto

Relações:

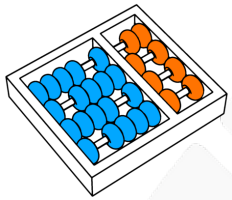
- **s1.isdisjoint(s2)**: são disjuntos?



Métodos úteis de conjunto

Relações:

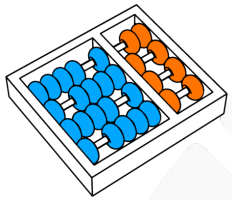
- **s1.isdisjoint(s2)**: são disjuntos?
- **s1.issubset(s2)**: **s1** é subconjunto de **s2**?



Métodos úteis de conjunto

Relações:

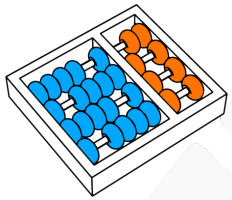
- **s1.isdisjoint(s2)**: são disjuntos?
- **s1.issubset(s2)**: **s1** é subconjunto de **s2**?
  - Ou **s1 <= s2**.



Métodos úteis de conjunto

Relações:

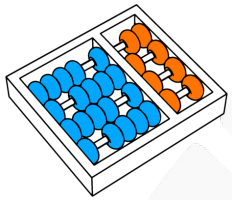
- **s1.isdisjoint(s2)**: são disjuntos?
- **s1.issubset(s2)**: **s1** é subconjunto de **s2**?
  - Ou **s1 <= s2**.
- **s1.issuperset(s2)**: **s1** é superconjunto de **s2**?



Métodos úteis de conjunto

Relações:

- **s1.isdisjoint(s2)**: são disjuntos?
- **s1.issubset(s2)**: **s1** é subconjunto de **s2**?
  - Ou **s1 <= s2**.
- **s1.issuperset(s2)**: **s1** é superconjunto de **s2**?
  - Ou **s1 >= s2**.



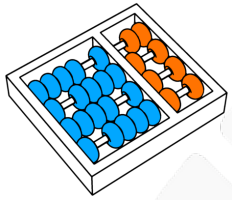
Métodos úteis de conjunto

Relações:

- **s1.isdisjoint(s2)**: são disjuntos?
- **s1.issubset(s2)**: **s1** é subconjunto de **s2**?
  - Ou **s1 <= s2**.
- **s1.issuperset(s2)**: **s1** é superconjunto de **s2**?
  - Ou **s1 >= s2**.

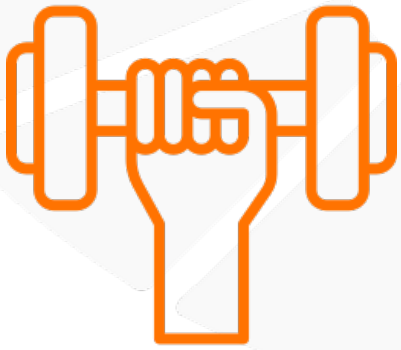
Como sempre, veja a documentação para mais métodos e detalhes!

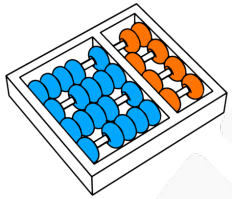
Exercícios



Conjuntos

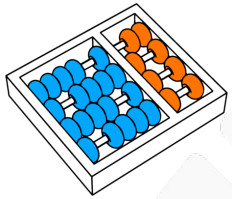
Vamos fazer alguns exercícios?





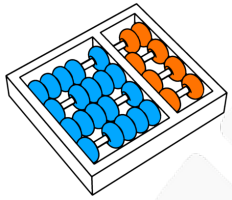
Exercícios

1. Faça um programa que encontre o mínimo e o máximo em um dado conjunto de números.



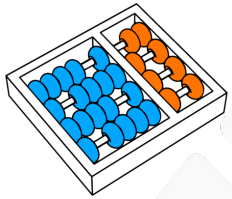
Exercícios

1. Faça um programa que encontre o mínimo e o máximo em um dado conjunto de números.
2. Faça uma função que dados dois conjuntos de números inteiros, retorna dois novos conjuntos: um com os dobros dos elementos repetidos e outro com a metade (inteira) dos elementos não repetidos.



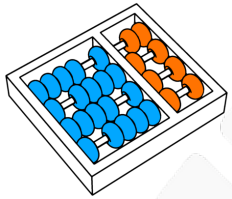
Exercícios

1. Faça um programa que encontre o mínimo e o máximo em um dado conjunto de números.
2. Faça uma função que dados dois conjuntos de números inteiros, retorna dois novos conjuntos: um com os dobros dos elementos repetidos e outro com a metade (inteira) dos elementos não repetidos.
3. Faça uma função que dados dois conjuntos, retorna um conjunto contendo qualquer valor que pode ser obtido como soma de um elemento do primeiro com um elemento do segundo.



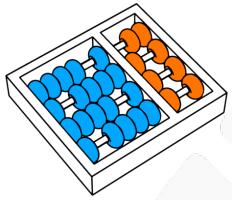
Exercícios

1. Faça um programa que encontre o mínimo e o máximo em um dado conjunto de números.
2. Faça uma função que dados dois conjuntos de números inteiros, retorna dois novos conjuntos: um com os dobros dos elementos repetidos e outro com a metade (inteira) dos elementos não repetidos.
3. Faça uma função que dados dois conjuntos, retorna um conjunto contendo qualquer valor que pode ser obtido como soma de um elemento do primeiro com um elemento do segundo.
4. Faça uma função que recebe um conjunto de números e retorna o maior produto entre dois números do conjunto.



Exercícios

1. Faça um programa que encontre o mínimo e o máximo em um dado conjunto de números.
2. Faça uma função que dados dois conjuntos de números inteiros, retorna dois novos conjuntos: um com os dobros dos elementos repetidos e outro com a metade (inteira) dos elementos não repetidos.
3. Faça uma função que dados dois conjuntos, retorna um conjunto contendo qualquer valor que pode ser obtido como soma de um elemento do primeiro com um elemento do segundo.
4. Faça uma função que recebe um conjunto de números e retorna o maior produto entre dois números do conjunto.
5. Faça uma função que recebe um conjunto de inteiros **S** e um inteiro **x** e retorna um conjunto contendo todos os pares de valores de **S** que somados resultam em **x**.



Exercícios

1. Faça um programa que encontre o mínimo e o máximo em um dado conjunto de números.
2. Faça uma função que dados dois conjuntos de números inteiros, retorna dois novos conjuntos: um com os dobros dos elementos repetidos e outro com a metade (inteira) dos elementos não repetidos.
3. Faça uma função que dados dois conjuntos, retorna um conjunto contendo qualquer valor que pode ser obtido como soma de um elemento do primeiro com um elemento do segundo.
4. Faça uma função que recebe um conjunto de números e retorna o maior produto entre dois números do conjunto.
5. Faça uma função que recebe um conjunto de inteiros **S** e um inteiro **x** e retorna um conjunto contendo todos os pares de valores de **S** que somados resultam em **x**.
6. Faça uma função que recebe um conjunto de strings e retorna o maior prefixo de todas as strings do conjunto.

Dicionários e conjuntos

MC102 - Algoritmos e Programação de Computadores

04/25

14

Santiago Valdés Ravelo
https://ic.unicamp.br/~santiago/
ravelo@unicamp.br

UNICAMP